Representation Ensemble Learning Applied to Facial Expression Recognition

Bruna Rossetto Delazeri[1*†], Andre Gustavo Hochuli[1†], Jean Paul Barddal[1†], Alessandro Lameiras Koerich[2†], Alceu de Souza Britto Jr.[1,3†]

[1]Graduate Program in Informatics (PPGIA), Pontifícia Universidade Católica do Paraná (PUCPR), Imaculada Conceição, Curitiba, 80215-901, Paraná, Brazil.

[2]Department of Software and IT Engineering, École de Technologie Supérieure (ÉTS) - University of Québec, 1100 Notre-Dame St W, Montreal, H3C 1K3, Quebec, Canada.

[3]Postgraduate Program in Applied Computing, State University of Ponta Grossa (UEPG), Av. General Carlos Cavalcanti, Ponta Grossa, 84010-330, Paraná, Brazil.

*Corresponding author(s). E-mail(s): bruna.delazeri@pucpr.edu.br;
Contributing authors: aghochuli@ppgia.pucpr.br; jean.barddal@ppgia.pucpr.br; alessandro.koerich@etsmtl.ca; alceu@ppgia.pucpr.br;
†These authors contributed equally to this work.

Abstract

This work introduces the representation ensemble learning algorithm, a novel approach for generating diverse unsupervised representations rooted in the principles of self-taught learning. The ensemble comprises convolutional autoencoders (CAEs) learned in an unsupervised manner, fostering diversity via a loss function designed to penalize similar CAEs' latent representations. We employ support vector machines, bagging, and random forest as primary classification methods for the final classification step. Additionally, we incorporate KnoraU, a well-established technique used to dynamically select competent classifiers based on a test sample. We evaluate various fusion strategies, including sum, product, and stacking, to comprehensively assess the ensemble's performance. A robust experimental protocol considering the facial expression recognition problem shows that

the proposed approach based on self-taught learning surpasses the accuracy of fine-tuned convolutional neural network (CNN) models. In terms of accuracy, the proposed method is up to 9.9 and 6.3 percentage points better than the CNN-based models fine-tuned for JAFFE and CK+ datasets, respectively.

Keywords: Self-taught Learning, Facial Expression Recognition, Convolutional Neural Network, Autoencoder

1 Introduction

The study of facial expression recognition (FER) has been ongoing for over two decades, as it conveys emotional states during human communication that play an essential role in society. Moreover, recognizing emotions has several applications, some of them with substantial social impact, such as identifying the autism spectrum [1] and chronic depression disorders [2], or monitoring driver fatigue in safety car systems [3].

The main challenges on the FER task are related to the different environmental conditions like lighting variations, non-frontal image acquisition, low-intensity face expression, and differences in facial expressions among gender, culture, and age groups [4]. Additionally, the scarcity of training data concerning its quantity and quality (samples of facial emotions) for some specific FER applications makes a full supervised-based approach bothersome [5]. In other words, a significant limitation of purely supervised systems is their dependence on large volumes of labeled data. Labeling is, in some FER applications, a costly and time-consuming process. The deep learning-based methods disseminated with convolutional neural network (CNN) models have made such a limitation even more evident [6].

An up-and-coming alternative to mitigate the problem related to the lack of labeled data is transfer learning (TL). A particular case of TL, named self-taught learning (STL) [7], consists of learning a representation (feature extractor) from unlabeled data, which is later applied to a target domain that is potentially from a different distribution. The main idea behind STL relies on natural human learning where unlabeled data is considered an essential foundation for high-level learning, being responsible for providing a more significant discriminating power [7]. Even with enough labeled data, STL can enhance the learning process, as observed in the experiments conducted in [8].

The combination of extraction and classification techniques is essential for FER systems. The main challenges in efficient facial expression recognition are the selection of efficient feature extraction and classification techniques. Handcrafted features, such as local binary patterns (LBP) [9, 10], scale-invariant feature transforms (SIFT) [11], gray level co-occurrence matrix (GLCM) [12], sped up robust features (SURF) [13] and histograms of oriented gradient (HOG) [14] achieved a breakthrough in various fields. Although using different descriptors to extract handcrafted features is possible, it is currently common to choose to learn the representation, removing the task of defining the features from the developer.

The potential of deep learning approaches to generate highly effective data representations has been widely recognized. In FER research, solutions based on CNNs have gained prominence [15]. Similarly, the convolutional autoencoder (CAE) is a promising strategy to make unsupervised-based solutions [16, 17] possible. In our previous study [18], we noticed that unsupervised learned representations yielded an effective facial emotion recognition method. Built upon such prior work, we introduce in this paper a new representation learning ensemble (REL) algorithm to produce a pool of representations employing STL principles. The main idea behind REL is to explore various strategies to learn representations (feature extractors) unsupervised and consider a custom loss function designed to promote diversity. Moreover, in the context of the FER problem, we evaluate how competitive the REL results are compared to CNN-based architectures trained in a supervised approach. We also compare our method with the current state-of-the-art in a common experimental protocol.

Two research questions guide this paper. The first, referred to as RQ1, concerns generating a diverse pool of unsupervised representations: "Does using a pool of unsupervised representations generated by the REL algorithm contribute to the performance of FER solutions?" However, we believe that more important than providing a new STL-based method to generate a pool of complementary representations is showing how far its performance is from that of supervised-based solutions. Thus, our second research question (RQ2) is "How does the new REL algorithm compare to supervised CNN-based architectures regarding facial expression recognition performance?"

By answering these research questions, the present manuscript provides a two-fold contribution. The first contribution is a new algorithm to generate unsupervised representations using STL focusing on diversity. More specifically, CAEs are trained considering different diversity induction strategies, i.e., distinct model initialization, architectures, and training data. The rationale is to investigate if complementary unsupervised learned representations can contribute to the performance of a FER solution. The second contribution is a robust performance comparison between the proposed unsupervised FER solution and supervised strategies under the same conditions, i.e., the same experimental protocol, and considering an authentic and demanding task like FER.

A robust experimental protocol considering three auxiliary datasets (Kyoto, LFW, and LabelMe) necessary for the unsupervised generation of representations proposed here and two well-known FER datasets (Jaffe and CK+) has shown that the proposed method may surpass the accuracy of CNN-based solutions, including two fine-tuning strategies applied to ten different CNN architectures and the training of two CNNs from scratch. Furthermore, the proposed FER framework demonstrated superiority over several studies recently published in high-impact journals, covering both deep learning approaches and handcrafted frameworks and following the leave-one-subject-out (LOSO) cross-validation protocol.

This paper is organized as follows. Section 2 introduces related works. Section 3 describes the proposed algorithm for learning a pool of diverse representations, while Section 4 presents the experiments undertaken to answer our research questions. Finally, Section 5 draws our conclusions and directions for future work.

2 Related Works

Various techniques have been proposed as FER solutions in the last few decades. Ensemble learning has become widely utilized in emotion recognition owing to its superior accuracy and generalization. Exploring diversity to create ensembles of CNNs, Renda et al. [19] investigated different strategies for inducing diversity in an ensemble of CNNs applied to FER. The results on the FER2013 dataset showed that seed variation yielded the best recognition results, while variations on the pre-training process of their CNNs achieved the best run-time performance.

A novel method for FER using an ensemble of CNNs with probability-based fusion was proposed by Wen et al. [20]. The ensemble is constructed by randomly varying parameters and architecture around the optimal values for CNN. Each CNN is trained to output a probability for each class, which is then fused using probability-based fusion. Experiments were conducted using 100 CNNs, and the best average results were achieved when the ensemble size was 35 for all datasets. The method was tested on benchmark datasets, achieved 50.70% and 76.05% for JAFFE and CK+ datasets, respectively, and outperformed other compared methods in terms of accuracy.

An ensemble learning method using electroencephalogram (EEG) signals is proposed by Li et al. [21]. The method uses a sliding time window to extract features and L1 regularization to select effective features. It then applies a model selection method to choose the best basic analysis sub-models and an ensemble operator to convert classification results. The optimal parameters are determined using multiple objective particle swarm optimization. The method is evaluated on two public datasets (DEAP and SEED) using the LOSO experimental protocol. The average accuracy rates for arousal and valence are 65.70% and 64.22%, respectively, on the DEAP dataset, and the average accuracy on the SEED dataset is 84.44%.

TL has gained popularity and emerged as a promising area in machine learning due to its broad potential applications and has been extensively applied to computer vision tasks. Dhankhar et al. [22] introduced ResNet50 and VGG16 architectures for facial emotion recognition and proposed an ensemble that combines both CNN models. The ensemble model outperformed the baseline SVM as well as the individual ResNet50 and VGG16 networks, achieving the highest accuracy of 75.8%. The SVM had an accuracy of 37.9%, while ResNet50 and VGG16 had accuracy rates of 73.8% and 71.4%, respectively. The authors also explored TL for FER using pre-trained AlexNet, VGG, and ResNet architecture networks. They achieved an average accuracy of 90% on the combined JAFFE and CK+ datasets.

Chowdary et al. [23] discuss using TL techniques in the FER scenario. Pre-trained networks such as ResNet50, VGG19, Inception V3, and MobileNet are employed. The fully connected layers of these pre-trained CNNs are removed and replaced with their own fully connected layers suitable to the specific task requirements. These newly added layers are then trained to update the weights. The experiment was performed using the CK+ dataset, resulting in an average accuracy of 96%.

A novel pipeline strategy that gradually improves the accuracy of FER by first training the dense layer(s) and then tuning each pre-trained CNN block successively was proposed by Akhand et al. [24]. The proposed FER system is tested on eight pre-trained CNN models using the KDEF and JAFFE facial image datasets. FER is

challenging even for frontal views and is further complicated by the diversity of profile views in the KDEF dataset. The proposed method achieved remarkable accuracy on both datasets with pre-trained models, achieving a FER accuracy of 96.51% and 99.52% on KDEF and JAFFE datasets, respectively, on a 10-fold cross-validation way.

Proposed by Raina et al. [7], the STL framework is a particular case of TL that uses unlabeled data from distributions other than of the problem at hand to learn a high-level representation in an unsupervised manner. This approach is motivated by the lack of training instances in many applications, and authors argue that a robust representation can be obtained from unlabeled samples, such as low-level discriminating structures in natural images, such as corners, curves, and shapes [25]. As a consequence, several authors have employed STL in diverse classification tasks, including audio [26], text [27, 28], image [29–31] and sensor data [32].

Inspired by STL to mitigate the problem related to the lack of samples for training in some FER applications and the use of ensembles to provide a way to generate a pool of diverse representations, we present our proposed algorithm algorithm in the next section.

3 Proposed Approach

The novelty in the proposed approach to solving the FER problem is an algorithm to generate a pool of representations inspired by STL and diversity concepts. Our approach is called representation ensemble learning (REL). REL has two essential features to promote diversity: i) it explores different strategies to generate diverse CAE architectures; ii) it uses a custom loss function to train different CAEs and promote diversity amongst latent representations.

Fig. 1 presents a general overview of the complete STL strategy adopted in this paper. In the **unsupervised representation learning step (Step 1)**, a high-level representation is learned using unlabeled data. This is the part in which our proposal is novel. Next, in the **feature building step (Step 2)**, feature vectors are extracted from the labeled data of the target domain using the representation learned in the previous step. Finally, in the **supervised learning step (Step 3)**, the feature vectors extracted in Step 2 are used to train a FER classification model (either using monolithic or ensemble-based classifiers).

3.1 Strategies Explored to Generate Diversity

We observe in the literature robust evidence that greater diversity is highly correlated with the increase in supervised CNN-based ensemble accuracy [19, 33]. Consequently, we follow the rationale of inducing diversity when creating the pool of unsupervised representations as illustrated in Fig. 2. The proposed algorithm uses a CAE to automatically learn meaningful representations from raw data, eliminating the need for manual feature engineering. The different strategies to vary specific CAE parameters are as follows:

- **Random Seed (S)**: here, different representations are generated by varying the distribution of weights during the CAE initialization process. The architecture may

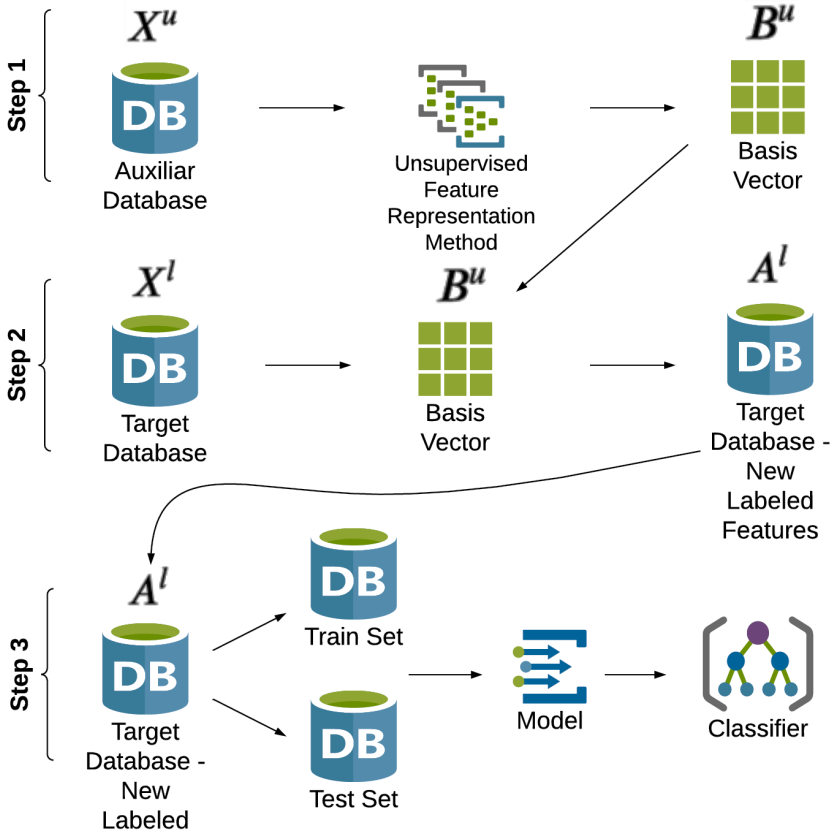

Fig. 1 STL steps: (1) Unsupervised representation learning; (2) Feature building; (3) Supervised learning.

be the same from one CAE to another, but the seed of the initialization process differs. The input is the number of representations (R) generated with different seeds. The algorithm randomly selects a seed at each generated CAE in the $[0, 1000]$ interval.

- **CAE Architecture (A)**: here, we explore representation generation using different CAE architectures. One must define the network depth (number of layers, D), the filters used, and the dimensionality (I) of the intermediate (latent) layer. The generator will create the first architecture with a depth equal to the number of defined layers (D), then the second will have $(D-1)$ up to the last architecture with depth $D = 1$, which has one input, one intermediate, and one output layer (basic structure of a CAE). The representations generated here differ in depth on the CAE architectures but use the same number of neurons (I) in the latent layer.
- **Latent Representation (L)**: in this strategy, diversity is induced by varying the number of neurons in the middle layer of the CAE, named the latent layer. The

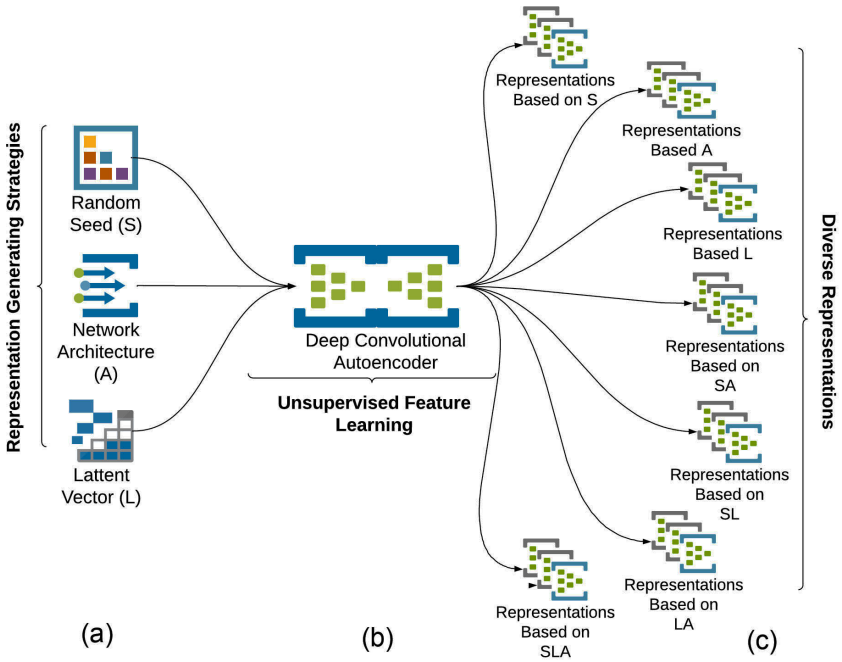

Fig. 2 Automatic unsupervised strategy generator for problem representation. (a) Different generating strategies of representations. (b) CAE for learning high-level representations. (c) Generated representations based on strategies: seeds (S), CAE architecture (A), latent representation (L), seeds + CAE architecture (SA), seeds + latent representation (SL), latent representation + CAE architecture (LA), and seeds + latent representation + CAE architecture (SLA).

input is the number of representations (R) to be generated with different dimensionality for the latent layer. The algorithm randomly selects a dimensionality from the $[150, 2500]$ range at each iteration.

The aforementioned strategies to generate diversity can also be combined as follows:

- **SA**: it combines the strategy based on seeds (S) and CAE architecture (A). The representations generated by this combination have variations in CAE initialization and architecture (number of layers).
- **SL**: it explores the combination of seeds (S) and latent representation (L) strategies, where the variation of both CAE initialization and latent layer dimensionality are adopted for each representation generated.
- **LA**: it explores the use of latent representation (L) and CAE architecture (A). Here, we use two approaches directly related to the CAE architecture, varying the dimensionality of the latent layer and the CAE depth in each representation generated.
- **SLA**: in the last combination, we consider all the strategies seeds (S), latent representation (L), and CAE architecture (A) for each representation generated.

3.2 Custom Loss Function

The loss function used in the REL algorithm contains two terms: a general loss term and a penalty term. The first term is the mean squared error (MSE) denoted in Equation 1:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{P}\sum_{i=1}^{P}(X_i - \hat{X}_i)^2 \tag{1}$$

where X_i is the original input data, $\hat{X}_i$ is the output data reconstructed by the CAE, and P is the total number of instances according to the batch size used to train the CAE.

The MSE calculates the average of the squares of the discrepancies between the model's predictions ($\hat{X}_i$) and the actual values (X_i) for a dataset composed of P examples. Its primary purpose is to minimize this value, aiming to make the outputs ($\hat{X}_i$) as similar as possible to the original inputs (X_i). This reflects the central goal of the CAE in learning a more concise and informative representation of the inputs, where a smaller value indicates a more precise reconstruction of the inputs.

However, the MSE term alone cannot guarantee diversity between the generated representations inside the ensemble. Consequently, a custom penalty term is added to the loss function after generating the first representation to ensure diversity between representations. Since we want to maximize the difference between the latent representations of different CAEs, the penalty term is added to the loss function as a negative term. Equation 2 represents the proposed custom loss function:

$$\mathcal{L}_{\text{MSE}_\text{P}} = \frac{1}{P}\sum_{i=1}^{p}(X_i - \hat{X}_i)^2 - \beta\frac{1}{T}\sum_{k=1}^{T}(R_k^{\text{last}} - R_k^{\text{curr}})^2 \tag{2}$$

where β defines the penalty provided to the similarity between the output of the R_k^{last} and R_k^{curr}, i.e., the latent vectors generated by the current and previous CAE, which is averaged across a validation set comprised of T instances. The dimensionality of the latent vectors R_k^{last} and R_k^{curr} may differ due to the differences in the CAE architectures, accordingly with the strategy used to generate diversity in the pool. This is the case when varying the dimensionality of the latent representation is the used strategy. Thus, we applied the principal component analysis (PCA) to make both latent vectors have the same dimensionality to allow computing the penalty term in Equation 2. For this purpose, we consider the size of the smallest latent vector as the number of selected PCA components.

The greater the similarity between the latent vectors, the greater the penalty applied. The idea is to increase the diversity between the latent representations of a previous and a current CAE at each iteration. The need to adjust the β parameter emerges from the search for a crucial trade-off between the representations generated and the desired diversity. The sweet spot is where the quality of the representations produced is maximized while encouraging the desired diversity.

3.3 Representation Ensemble Learning (REL) Method

This section introduces the proposed REL method, which is described here by three algorithms. Algorithm 1 presents the primary function responsible for generating a pool of unsupervised representations. The input parameters of the REL function play a pivotal role in configuring and executing the proposed method. In particular, we have ψ representing the number of representations to be generated, `str` defining the strategy to create the pool (S, L, A, SL, SA, LA, or SLA), X_t^u as the auxiliary unsupervised training dataset, X_v^u as the additional unsupervised validation dataset, both used for training the created autoencoders. The parameter β is related to the proposed custom loss function, defined in Equation 2 and used in Algorithm 2. It is a constant value that directly influences the magnitude and direction of the loss term in our function with the idea of penalizing the generation of similar CAEs. Finally, the structure `params` contains the configuration parameters for a default CAE architecture, including the number of layers, filter sizes, activation functions, and other fundamental hyperparameters. The default CAE is used if `params` is `NULL`.

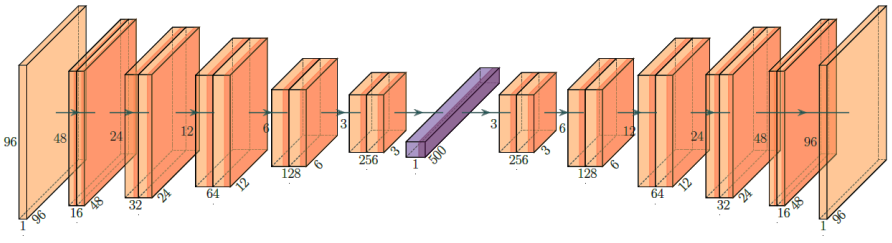

Fig. 3 Default CAE Architecture.

Fig. 3 shows the default CAE architecture used. It has input and output layers of dimensionality 96×96×1, a depth of 5, and a latent vector size equal to 500. However, depending on the strategy used to generate the pool, the latent vector dimension and the CAE depth may change at each iteration of the REL function, according to lines 9-13 and 14-20 of the Algorithm 1, respectively. All the parameter values of the default CAE are shown in Table 1. Such parameters were defined empirically based on preliminary experiments and iterative adjustments performed throughout the development of the proposed model. We have experimentally defined these CAE parameter values with FER performance in mind. It is worth noticing that the CAE can be customized to suit specific application requirements. Finally, the output parameter ϱ is a pool of feature extractors trained on an unsupervised approach.

At each iteration of the Algorithm 1, we create an encoder, a latent vector, and a decoder to compose a new CAE (see lines 21-28), which is compiled at line 29 considering the provided `Optimizer` and `Custom_Loss` parameters. Following that, on lines 30-34, the created CAE is trained. As one can see in line 33, from the second iteration, the output prediction (R_k^{last}) of the previous CAE estimated on the validation dataset (X_v^u) at line 35 is available to compute the customized loss function correctly. In line 36, we append the encoder in the pool as a new member, a feature extractor.

Algorithm 1 Representation Ensemble Learning (REL)

Input: ψ, as the number of representations,
X_t^u, as the auxiliary train dataset,
X_v^u, as the auxiliary validation dataset,
`params`, as the default CAE parameters (Table 1),
β, as a constant for loss penalization term,
`str`, as the strategy to vary CAE parameters
Output: ϱ, as the pool of diverse representations

function REL(ψ, `str`, X_t^u, X_v^u, β, `params`)
  `max_depth` ← `params.CAE_Depth`
  `latent_size` ← `params.Latent_Size`
  `seed_number` ← `params.Seed`
  for k **in** [1..ψ] **do**
    if (`str` in [S, SL, SA, SLA]) **then**
      `seed_number` ← `random(range(`0 : `params.Max_random_seed))`
    end if
    if `str` in [L, SL, LA, SLA] **then**
      ini ← `params.Min_latent_size`
      end ← `params.Max_latent_size`
      `latent_size` ← `random(range(`*ini* : *end*))
    end if
    if (`str` in [A, SA, LA, SLA]) **then**
      if (`max_depth`= 1) **then**
        `max_depth` ← `params.CAE_Depth`
      end if
    else
      `max_depth` ← `max_depth` −1
    end if
    for i **in** [1..`max_depth`] **do**
      `encoder.create_convolutional_layer(params)`
    end for
    `encoder.create_flatten_layer(params)`
    for i **in** [1..`max_depth`] **do**
      `decoder.create_transpose_layer(params)`
    end for
    `CAE.(encoder,decoder)` ▷ CAE model constructor
    `CAE.compile(params.Optimizer, params.Custom_Loss)`
    if $k = 1$ **then**
      `CAE.fit(params.Epoch, params.Batch_Size,` X_t^u, X_v^u, `null)`
    else
      `CAE.fit(params.Epoch, params.Batch_Size,` $X_t^u, X_v^u, R_k^{\text{last}}$)
    end if
    R_k^{last} ← `CAE.encoder.predict(`X_v^u) ▷ Prediction on validation set
    ϱ`.append(CAE.encoder)`
    `delete CAE`
  end for
  return ϱ
end function

Table 1 Default parameter values for the CAE model.

Parameter	Value(s)
Stride	2
Activation	RELU
Kernel_Size	3
Filters	[16, 32, 64, 128]
Output_Activation	Linear
Input_Size_Data	[96, 96, 1]
Latent_Size	500
Seed	42
CAE_Depth	5
Custom_Loss	Custom Loss (Equation 2)
Optimizer	SGD
Epoch	20
Batch_Size	60
Padding	Same
Max_random_seed	10000
Min_latent_size	150
Max_latent_size	2500

We detail the `Custom_Loss` function in the Algorithm 2. As one can see in line 2 of this algorithm, in the first iteration, REL uses a regular loss denoted on the Equation 1. However, for the second representation and so on, the cost function used is the customized one, indicated in Equation 2. When training a new representation, we try to diversify regarding the one created in the last REL iteration.

In line 5 of Algorithm 2, the encoder corresponding to the current representation extracts the features from the validation set X_v^u. This is done using the predict function, and the result obtained is stored in R_k^{curr}. If we opt for the latent vector strategy to introduce diversity in the representations, it is essential to highlight that the dimensionality of the intermediate layer will vary between them. Therefore, for us to correctly calculate the difference between R_k^{curr} and R_k^{last}, it is essential to equalize their dimensions. Thus, in line 6, the smallest value between these dimensions is obtained and used as the number of components to be considered in the PCA, as indicated in lines 7 and 8. This process of equalizing the dimensions is crucial for guaranteeing the consistency and comparability of representations throughout the algorithm.

In line 9 of Algorithm 2, we calculate the sum of the quadratic differences between the last generated and current representations. The result of this calculation is then used, on line 10, as part of the penalty term, multiplying the value of β. This process is essential to adjust the impact of the penalty on the cost function and ensure that the difference between representations is considered according to the regularization parameter β. Such a regularization parameter reflects our concern about generating similar representations.

After completing the training of all representations as outlined in the Algorithm 1, we move on to the next step, called "Feature Building" (Step 2 of Fig. 1), which extracts feature vectors from the labeled data X^l using the weights of each previously trained `CAE.encoder` available at the pool ϱ. We detail this process in the Algorithm 3, which has the labeled target dataset X^l and the pool of feature extractors ϱ as

Algorithm 2 Custom Loss Function - Eq. 2

Input: β, as the constant value for the loss function,
X_t^u, as the auxiliary train dataset,
X_v^u, as the auxiliary validation dataset,
R_k^{last}, as the output of the latent layer of the last CAE,
X_i, as the true pattern,
$\hat{X}_i$, as the output pattern

Output: `loss_value`, as the computed loss value

function CUSTOM_LOSS(β , X_t^u, X_v^u, R_k^{last})
  if (R_k^{last} = `NULL`) **then**
    `loss_value` $\leftarrow \frac{1}{P}\sum_{i=1}^{P}(X_i - \hat{X}_i)^2$ (Equation 1)
  else
    $R_k^{\text{curr}} \leftarrow$ `CAE.encoder.predict`(X_v^u)
    `n_components` $\leftarrow \min($`length`(R_k^{last}), `length`(R_k^{curr}))
    `PCA_curr` $\leftarrow$ `PCA(n_components).fit_transform`(R_k^{last})
    `PCA_last` $\leftarrow$ `PCA(n_components).fit_transform`(R_k^{curr})
    `dif` $\leftarrow \sum_{k=1}^{T}$ (`PCA_last` $-$ `PCA_curr`)^2
    `penalty_term` $\leftarrow \beta \times$ `dif`
    `loss_value` $\leftarrow \frac{1}{P}\sum_{i=1}^{P}(X_i - \hat{X}_i)^2 -$ `penalty_term` (Equation 2)
  end if
  return `loss_value`
end function

input. The output is a pool of feature sets A^l considered a novel way to represent X^l. Such a new representation of the target dataset can subsequently be employed in constructing supervised classifiers.

Algorithm 3 Feature Building Step 1

Input: X^l as the target dataset,
ϱ as feature extractor pool

Output: A^l as a pool of new labeled features of the target dataset

function FEATURE_EXTRACTION(X^l , ϱ)
  $A^l \leftarrow []$
  for fe **in** [ϱ] **do**
    `feature_set` $\leftarrow$ `fe.predict`(X^l)
    A^l`.append(feature_set)`
  end for
  return A^l
end function

The supervised model training phase represents the final step of the self-taught learning process. In this step, we evaluate the performance of a diverse set of classifiers, including support vector machines (SVM), random forest (RF), bagging (BG) with

decision tree as a base classifier, and Knora union-based dynamic selection (KnoraU) in the context of decision trees (DT) and RF ensembles. These classifiers were examined to determine their effectiveness in leveraging the previously generated representations. The evaluated fusion strategies include sum, product, and stacking. Table 2 presents the parameters empirically defined for each technique evaluated in the last step of the proposed method. The parameters not present in that table use the default values available in the Scikit-learn framework [34]. We conducted all experiments considering the same configuration, ensuring the reproducibility of the observed results.

Table 2 Parameter settings of algorithms used in Step 3. Classifiers: single (SVM), ensembles (BG: bagging with decision tree as a base classifier, RF: random forest), KnoraU (DT: pool of trees; RF: random forest).

Algorithm	**Parameters**	
SVM	Kernel	Linear
	Penalty Parameter (C)	1e-6
	Class Weight	Balanced
	Probability	True
BG with DT	Max Depth	10
	Tree Max Features	sqrt
	Number of Base Estimators	100
	% of Training Samples	1.0
RF	Max Depth	10
	Number of Trees	100
	Oob_score	True
KnoraU	Number of neighbors	7
	pool_classifiers	Bagging DT RF
Stacking	Meta classifier	Logistic Regression
	Solver	`lbfgs`
	Penalty parameter C	0.1

4 Experiments and Discussion

This section describes the experiments to evaluate the proposed REL method, answering our two research questions.

4.1 Experimental Protocol

The target datasets are the Japanese Female Facial Expression (JAFFE) [35] and the Extended Cohn-Kanade (CK+) [36]. The JAFFE dataset is a laboratory-controlled image dataset with 213 images of 10 subjects (Japanese female models), with six basic facial expressions (happiness, anger, disgust, fear, sadness, and surprise) and a neutral one. In the JAFFE dataset, all ten subjects have one or more images for each class. The CK+ dataset is a laboratory-controlled dataset widely used to evaluate FER systems. It contains 523 sequences from 123 subjects, and 327 images of 118 subjects

are labeled with seven basic facial expressions (anger, contempt, disgust, fear, joy, sadness, and surprise). In the CK+ dataset, not all subjects have images in all classes.

Before extracting features from face images, we pre-processed them, cropping only the region of the region of interest (face) and selecting the reference points. Fig. 4 illustrates the pre-processing applied to a sample image from JAFFE and CK+ datasets. To detect and crop the face area of the image, we use the Viola-Jones face detection method [37]. The reference points on the cropped image align the images so they are in the same position.

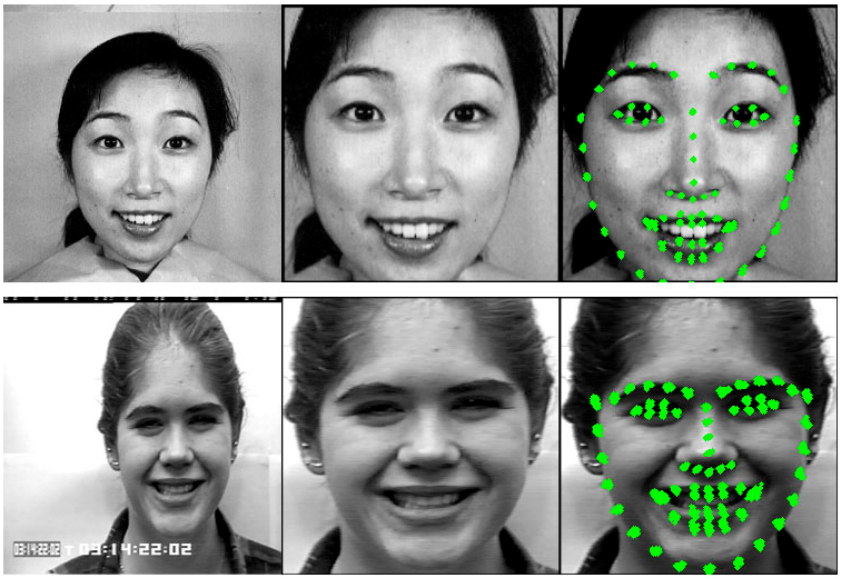

Fig. 4 Example illustrating the original image (left), face detection (middle), and landmark extraction (right) for (a) the JAFFE dataset and (b) the CK+ dataset.

As mentioned before, an unlabeled auxiliary dataset is necessary. The following auxiliary datasets were considered in our experiments:

- Kyoto Natural Images [38]: The dataset contains 62 natural images of resolution 256×200 pixels, used by several other works that apply the concepts of STL, thanks to its large variability. The images from our experimental protocol belong to a domain (FER) far from that dataset.
- LabelMe [39]: The dataset consists of 50,000 images (40,000 for training and 10,000 for testing), and each image is 256×256 pixels in size. The images belong to one of the 12 object classes: person, car, building, window, tree, sign, door, bookshelf, chair, table, keyboard, and head. For this paper, we used only the testing base.
- Labelled Face in the Wild (LFW) [40]: is a publicly available dataset of face photographs used for verification, also known as peer matching. The dataset has 13,233 images of faces collected on the web from 5,749 people.

The first two auxiliary datasets have images that do not belong to the FER domain but differ in size (62 images for Kyoto and 10,000 for LabelMe). The rationale is to show whether the size of the auxiliary dataset may impact the proposed method. On the contrary, the third dataset comprises faces closer to the target domain. Fig. 5 shows four image samples of each auxiliary dataset used in our experiments.

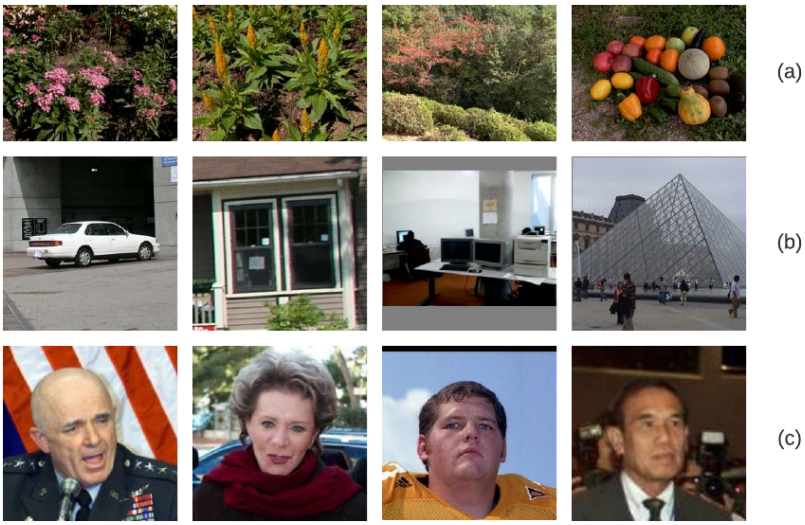

Fig. 5 Samples from the datasets (a) Kyoto Natural Images, (b) LabelMe, and (c) Labelled Face in The Wild.

The experimental protocol employs a LOSO cross-validation in the last step of the proposed method. In this cross-validation approach, the dataset is partitioned so that no subject in the test set appears in the training set. Suppose we have a dataset with N subjects; therefore, for each i-th fold, one subject will be designated as a test and the others for training. In such a protocol, when using the KnoraU dynamic ensemble selection, a subset of data (validation set) is extracted from the training set. Such a validation set is necessary since KnoraU needs to compute the competence of each classifier.

4.2 Results on the FER problem

For each target dataset (JAFFE and CK+), we performed experiments considering the three auxiliary datasets (Kyoto, LabelMe, and LFW) and representation ensembles of different sizes (5, 10, 15, 20, 30, 40, 50, 70, 100, and 150). The strategies S, A, L, SA, SL, LA, and SLA were tested in each experiment. It is crucial to notice that strategy A was used uniquely in the investigation, where we generated just five representations because it explores the depth of the CAE architecture. Thus, using a depth greater than

five layers makes the experiment almost impractical on the computational resources available for running the experiments.

At the last step of our STL-based approach, we used the following inducers: SVM, RF, BG, and dynamic ensemble selection based on KnoraU executed in a pool of DTs and RF. For each ensemble of representations, we present the result of the best representation and their fusion (sum, product, and stacking). Furthermore, the standard deviation of all models inside the ensemble is calculated to provide an idea of the variability achieved in the generated ensemble.

4.2.1 JAFFE as Target Dataset

Tables 3, 4, and 5 present the results for the JAFFE dataset when CAE is trained with Kyoto, LabelMe, and LFW datasets respectively. The best outcome for the JAFFE datasets was achieved with an ensemble of 50 representations trained on the Kyoto dataset, varying the latent vector dimensionality (strategy L), and using KnoraU dynamic ensemble selection executed on an RF at each representation. Thus, by combining the results of each representation using stacking, we achieved 66.66% accuracy. The standard deviation inside the corresponding pool of representations was (+/-) 3.33 percentage points. It is also important to note that the best single representation using KnoraU provided 42.00% accuracy.

When using the LabelMe dataset (Table 4), the best result was achieved with an ensemble composed of 50 representations varying the latent vector size, architecture, and seeds (strategy SLA). The last step (supervised) uses KnoraU dynamic ensemble selection executed on an RF at each representation. The best single representation provided 44.67% of accuracy, while by combining the results of each representation using stacking, we achieved 65.25%.

When using the LFW dataset (Table 5), the best result was achieved with an ensemble composed of ten representations. The best accuracy (65.25%) was observed by varying the seeds and the CAE architecture (strategy SA). The final inducer is an ensemble of SVMs combined via product rule. The best single SVM inside the ensemble achieved 61.39% of accuracy.

Tables 3, 4 and 5 also present the accuracy standard deviation among generated representations (between brackets beside the best single result) that varies from 2.50 to 6.89 percentage points (4.02 in average), what was expected in our method. The result analysis indicates that after 50 representations, there is no significant improvement in performance. Such an observation shows that, as in an ensemble of classifiers, we must empirically define the number of representations in our pool for each problem. In other words, a larger ensemble does not always mean a better performance. For instance, Fig. 6 shows that considering the results produced with the Kyoto dataset, an ensemble with 50 members may provide the same result as an ensemble with 150 members, substantially reducing complexity.

4.2.2 CK+ as Target Dataset

Tables 6, 7, and 8 present the results for the CK+ dataset when CAE is trained with Kyoto, LabemMe, and LFW datasets, respectively. The best result for the CK+ dataset (92.66%) was achieved with an ensemble of 50 representations trained on the

Table 3 Accuracy results using the LOSO protocol with a pool of 50 representations generated by the proposed method. Kyoto and JAFFE are used as auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a collection of DTs and RF. Strategies S, L, and A represent different seeds, latent layer dimensions, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of SVMs	Ensembles		Dynamic Selection (KnoraU)	
			BG	RF	DT	RF
S	Best Single	58.50 (3.24)	55.31 (4.24)	40.76 (3.63)	56.67 (4.22)	45.18 (4.12)
	Sum	59.15	61.50	60.09	59.15	58.21
	Product	59.15	57.27	59.62	59.15	57.27
	Stacking	59.62	58.68	43.66	61.97	**63.38**
L	Best Single	60.42 (4.00)	51.37 (4.26)	44.74 (3.94)	51.89 (4.30)	42.00 (3.33)
	Sum	60.09	61.03	60.56	61.03	62.44
	Product	60.09	61.97	60.56	61.03	62.91
	Stacking	61.03	60.56	48.82	61.97	**66.66**
SA	Best Single	61.19 (3.19)	49.54 (4.05)	44.45 (3.60)	50.47 (4.14)	42.71 (3.57)
	Sum	61.97	61.32	62.91	59.62	61.50
	Product	61.97	60.56	**64.78**	60.09	61.97
	Stacking	60.56	59.62	53.05	60.56	63.84
SL	Best Single	61.47 (3.41)	52.63 (4.40)	43.50 (3.74)	51.72 (4.21)	44.38 (4.40)
	Sum	61.03	61.97	57.74	61.03	60.09
	Product	61.50	60.56	57.27	62.91	60.09
	Stacking	62.44	59.62	43.66	62.91	**63.84**
LA	Best Single	60.61 (3.38)	50.05 (3.38)	47.23 (4.45)	52.88 (3.52)	48.14 (4.90)
	Sum	60.56	61.03	61.97	61.03	**62.91**
	Product	60.56	60.09	62.44	61.50	**62.91**
	Stacking	60.03	61.97	51.17	61.50	61.50
SLA	Best Single	61.35 (3.37)	51.62 (4.36)	44.46 (4.36)	52.94 (4.11)	47.41 (4.20)
	Sum	61.50	62.91	59.15	**63.38**	59.15
	Product	61.50	61.97	59.62	61.97	59.62
	Stacking	61.50	58.68	48.35	60.09	62.44

Kyoto dataset, varying the seeds and latent vector dimensionality (strategy SL) and using the fusion based on the stacking of SVMs. It is also essential to note that the best result based on a single SVM classifier in this experiment was 91.29%.

When using the LabelMe dataset (Table 7), the best result of 91.74% was achieved with an ensemble composed of 30 representations and by varying the latent vector dimensionality (strategy L), using an ensemble of SVMs. The best result when using the LFW dataset as an auxiliary dataset (90.51%) was observed using the SL strategy and an ensemble of SVMs, as shown in Table 8.

Unlike the JAFFE dataset, ensemble methods (BG, RF) and dynamic selection (KnoraU) were less effective on the CK+ dataset. One possible explanation is the relatively small standard deviation in accuracy among the generated representations on CK+ compared to JAFFE. The standard deviation for CK+ ranged from 0.7 to 5.4 percentage points (average of 2.3), suggesting a lack of significant variation in the quality of the representations.

As observed in the previous experiments on the JAFFE dataset, the behavior is similar here, confirming that we must define the best ensemble size empirically. Fig. 7 shows that the best ensemble size was 50 when considering Kyoto and 70 when using

Table 4 Accuracy results using the LOSO protocol with a pool of 50 representations generated by the proposed method. LabelMe and JAFFE are used as auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a collection of DTs and RF. Strategies S, L, and A represent different seeds, latent layer dimensions, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of SVMs	Ensembles		Dinamic Selection (KnoraU)	
			BG	RF	DT	RF
S	Best Single	57.08 (3.32)	50.53 (4.19)	44.30 (5.23)	50.79 (4.52)	42.44 (4.77)
	Sum	55.86	61.97	59.15	61.03	61.03
	Product	57.27	61.03	59.62	**62.44**	60.56
	Stacking	56.33	59.15	50.70	**62.44**	**62.44**
L	Best Single	58.96 (5.56)	48.66 (4.36)	42.62 (3.84)	49.58 (4.30)	43.33 (3.59)
	Sum	59.15	56.33	58.21	56.33	57.27
	Product	58.68	56.33	58.21	55.86	56.80
	Stacking	58.68	55.86	42.25	58.21	**60.56**
SA	Best Single	61.36 (3.20)	49.58 (3.96)	43.97 (4.72)	49.06 (3.94)	44.30 (4.24)
	Sum	60.56	60.56	62.91	60.09	63.84
	Product	61.50	58.68	62.91	60.56	63.38
	Stacking	58.68	62.44	53.52	59.62	**64.78**
SL	Best Single	59.44 (2.98)	46.35 (3.11)	44.18 (3.90)	47.86 (3.12)	44.31 (3.54)
	Sum	58.21	61.50	60.09	59.62	59.15
	Product	57.74	55.86	60.56	59.62	59.15
	Stacking	59.62	59.15	46.94	**63.38**	62.44
LA	Best Single	60.40 (3.51)	50.82 (5.00)	45.58 (4.65)	48.72 (4.96)	43.41 (4.61)
	Sum	59.62	57.27	59.15	59.62	58.68
	Product	60.09	58.68	60.09	58.68	58.21
	Stacking	60.56	61.00	47.41	61.03	**62.44**
SLA	Best Single	60.78 (3.89)	50.15 (3.98)	44.12 (3.84)	52.52 (4.05)	44.67 (4.14)
	Sum	60.56	59.62	60.09	60.56	60.56
	Product	61.03	61.03	59.62	60.56	60.56
	Stacking	59.62	61.03	50.23	61.97	**65.25**

LabelMe and LFW datasets. It points out that the performance of the generated ensemble is relative to the capacity to cover the problem space with complementary models, which does not depend on the pool size but on the algorithm's ability to generate diverse representations.

4.3 Ablation Study

After generating the first representation, we consider a penalty term in the cost function to ensure that subsequent representations are diverse, as denoted in Equation 2. The β parameter is a user-defined constant that plays an essential role in this scenario, as it is used to control the penalty applied for generating diversity between the representations. In other words, it regulates the importance of the penalty applied to representations to encourage the exploration of different regions of the feature space. The idea is to generate different autoencoders at each training step. We opted for a uniform β in this work, which proved effective and kept the model simple and efficient. However, different strategies can be investigated, for instance, varying the value

Table 5 Accuracy considering the LOSO protocol and a pool with 10 representations generated with the proposed method. LFW and JAFFE are used as auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a collection of DTs and RF. Strategies S, L, and A represent different seeds, latent layer dimensions, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of SVMs	Ensembles		Dynamic Selection (KnoraU)	
			BG	RF	DT	RF
S	Best Single	60.52 (3.65)	44.99 (3.23)	38.93 (2.65)	42.72 (3.08)	38.73 (2.67)
	Sum	58.68	53.99	50.70	52.58	51.64
	Product	60.09	53.52	51.64	53.52	51.17
	Stacking	**60.56**	55.39	44.13	51.17	53.52
L	Best Single	60.56 (3.82)	48.04 (3.84)	47.26 (5.11)	49.42 (3.96)	47.26 (4.83)
	Sum	61.03	58.21	56.80	58.21	**61.97**
	Product	60.56	60.56	57.74	58.68	61.03
	Stacking	59.15	56.80	48.82	56.33	60.09
SA	Best Single	61.39 (2.99)	46.33 (5.04)	39.72 (3.35)	50.34 (4.12)	42.87 (3.16)
	Sum	64.31	58.21	58.21	57.27	57.74
	Product	**65.25**	58.68	60.09	57.27	58.68
	Stacking	63.38	58.21	51.17	54.92	53.05
SL	Best Single	58.53 (4.01)	42.44 (3.78)	39.54 (3.18)	42.72 (3.64)	40.44 (3.51)
	Sum	**60.09**	55.86	57.74	52.58	55.86
	Product	59.62	57.74	56.80	52.58	55.39
	Stacking	59.62	54.92	46.94	49.29	55.86
LA	Best Single	59.17 (2.50)	50.83 (4.86)	43.97 (4.87)	51.21 (5.31)	45.38 (5.22)
	Sum	58.68	57.27	56.33	56.33	54.92
	Product	**59.62**	58.21	56.33	57.74	54.46
	Stacking	59.15	58.68	52.11	56.33	57.74
SLA	Best Single	58.92 (3.02)	54.24 (6.89)	42.54 (5.32)	51.88 (6.38)	44.51 (4.66)
	Sum	61.97	56.80	59.15	54.92	58.21
	Product	**62.44**	54.92	57.27	55.86	57.27
	Stacking	61.97	56.33	52.58	53.05	58.21

of β across the training steps. Increasing β value may be a strategy to create more different autoencoders during training.

The best results for JAFFE and CK+ datasets are shown in Tables 3 and 6, respectively. For the ablation study, we used the best performance settings. Thus, we generated 50 representations with the unlabeled dataset Kyoto and applied different values for the β. Table 9 shows the performance of the obtained model for each value of β.

The best results on JAFFE and CK+ datasets were found using $\beta = 0.001$. When we increase or decrease the value of β, we reduce model performance. The β value is experimentally adjusted to find the right balance between the quality of the generated representations and the desired diversity. If the β is too low, diversity might not be encouraged enough, and representations might become too similar. On the other hand, if the β is too high, the penalty can completely overwhelm the objective, resulting in very diverse but potentially low-quality representations.

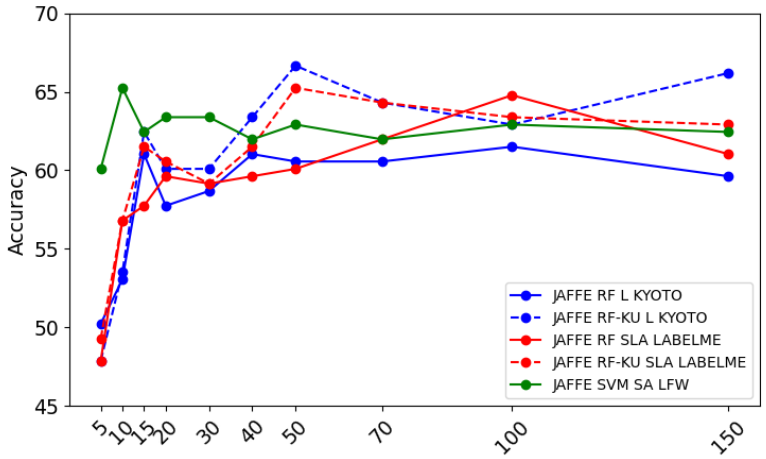

Fig. 6 The impact of varying the ensemble size in the best setup achieved for the JAFFE dataset when using Kyoto, LabelMe, and LFW datasets as auxiliary ones.

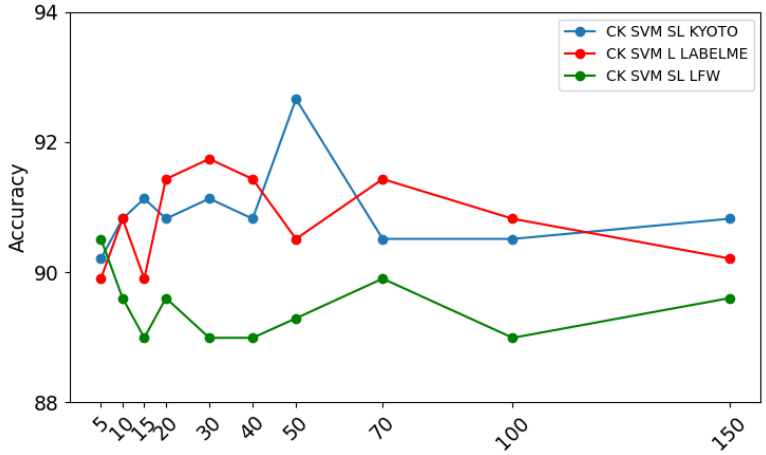

Fig. 7 The impact of varying the ensemble size in the best setup achieved for the CK+ dataset when using Kyoto, LabelMe, and LFW datasets as auxiliary ones.

4.4 Comparison with CNN-based Approaches and Related Works

As stated in Section 1, the performance of supervised approaches against STL is a matter of discussion once the former is widely used in the literature and generally achieves the best results. Thus, to answer our RQ2 correctly, we assess the traditional supervised learning strategy using the LOSO protocol presented in Section 4.1, in which the samples of test subjects are not present in the training step.

Table 6 Accuracy considering the LOSO protocol and a pool with 50 representations generated with the proposed method. Kyoto and CK+ are used as auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a pool of DTs and RF. Strategies S, L, and A represent different seeds, latent vector dimensions, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of SVMs	Ensembles		Dinamic Selection (KnoraU)	
			BG	RF	DT	RF
S	Single Best	88.48 (2.63)	71.75 (3.09)	65.09 (2.01)	71.90 (2.79)	66.62 (1.90)
	Sum	87.15	74.00	68.19	73.39	68.80
	Product	87.46	74.92	69.11	74.31	68.80
	Stacking	**89.90**	80.42	76.75	76.45	80.12
L	Single Best	89.87 (2.58)	72.45 (2.54)	69.78 (2.34)	71.10 (2.26)	69.13 (2.30)
	Sum	89.60	75.84	71.55	75.22	72.17
	Product	89.29	76.45	71.55	76.14	72.17
	Stacking	**90.82**	77.98	78.89	74.31	81.34
SA	Single Best	89.47 (1.80)	71.55 (2.47)	66.66 (1.90)	73.88 (2.60)	67.74 (1.81)
	Sum	89.90	74.92	70.64	75.22	72.17
	Product	89.29	77.37	70.64	75.84	72.17
	Stacking	**90.21**	80.12	75.84	75.84	78.89
SL	Single Best	91.29 (2.86)	72.57 (2.72)	68.31 (2.14)	72.14 (2.82)	67.37 (1.85)
	Sum	89.60	74.92	70.64	74.00	71.55
	Product	89.60	75.84	70.64	74.00	71.55
	Stacking	**92.66**	77.67	77.06	74.31	78.89
LA	Single Best	87.18 (2.24)	72.06 (2.72)	68.88 (2.02)	73.33 (2.85)	69.51 (2.24)
	Sum	86.85	77.37	71.55	77.06	72.47
	Product	87.46	77.37	72.17	77.06	72.47
	Stacking	**88.07**	79.81	76.14	78.28	79.81
SLA	Single Best	87.00 (3.05)	74.23 (3.16)	68.14 (2.20)	73.29 (3.04)	68.58 (2.28)
	Sum	87.76	76.45	71.86	75.53	71.86
	Product	87.76	77.06	71.86	76.14	72.17
	Stacking	**88.07**	79.51	76.45	76.45	79.81

In the state-of-the-art, there are a plethora of ready-to-use CNN architectures. To avoid a biased comparison, the rationale is to select well-known networks that provided a breakthrough in their architectures and achieved competitive performances on the ImageNet Challenge [41].

The first successful approach was made by Krizhevsky et al. [42], implementing a convolutional architecture named AlexNet composed of five layers. Next, Simonyan and Zisserman [43](VGG) proposed a more profound architecture. Contrary to the rationale of stacking layers that increase the depth of the network, Szegedy et al. [44] proposed the inception modules composed of a pattern of convolutional layers, pooling, and feature concatenation. The final architecture is a stack of Inception modules that provide a more comprehensive network and enhance the feature representation. So far, most of the proposed architectures have resorted to the fact that high performances should be achieved as the network grows. However, the vanishing of the gradient has become an issue. To minimize the overfitting, skip connections between adjacent layers were proposed in ResNet [45] and improved in DenseNet [46], which extends the skip connections between all network layers. Last, EfficientNet [47] proposed a technique to provide model scaling to improve the efficiency of the network.

Table 7 Accuracy considering the LOSO protocol and a pool with 30 representations generated with the proposed method. LabelMe and CK+ datasets are used as auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a collection of DTs and RF. Strategies S, L, and A represent different seeds, latent vector dimensions, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of SVMs	Ensembles		Dinamic Selection (KnoraU)	
			BG	RF	DT	RF
S	Best Single	89.57 (4.67)	69.50 (2.23)	66.93 (2.05)	69.97 (2.22)	66.72 (2.23)
	Sum	87.46	73.70	70.03	73.08	70.03
	Product	87.76	74.00	70.03	73.70	70.03
	Stacking	**89.90**	78.89	77.37	75.22	79.20
L	Best Single	90.38 (2.25)	72.16 (2.27)	65.43 (1.47)	71.99 (1.99)	66.76 (1.78)
	Sum	89.90	76.45	69.74	75.53	70.64
	Product	89.90	75.84	69.72	76.45	70.64
	Stacking	**91.74**	81.34	78.59	74.31	77.98
SA	Best Single	88.87 (1.91)	69.19 (2.13)	67.09 (2.11)	70.55 (2.16)	68.37 (2.09)
	Sum	88.68	74.92	70.33	75.53	70.33
	Product	88.68	76.75	70.64	75.53	70.33
	Stacking	**89.29**	79.81	75.22	77.98	77.37
SL	Best Single	89.71 (2.60)	71.08 (2.45)	69.03 (2.16)	71.51 (2.17)	67.31 (1.63)
	Sum	88.68	74.61	70.03	74.31	70.33
	Product	88.99	74.61	70.33	74.92	70.03
	Stacking	**90.51**	79.81	75.22	75.53	77.37
LA	Best Single	89.39 (3.84)	72.37 (2.68)	67.92 (2.53)	71.86 (2.82)	69.49 (3.01)
	Sum	90.21	77.06	70.94	75.84	70.94
	Product	89.90	77.67	71.25	76.75	70.94
	Stacking	**90.51**	80.12	75.53	77.98	79.51
SLA	Best Single	90.45 (2.38)	71.17 (3.21)	66.38 (2.36)	73.13 (3.43)	67.93 (1.95)
	Sum	89.29	75.22	70.33	74.92	70.64
	Product	89.29	74.92	71.25	75.22	70.64
	Stacking	**90.51**	79.81	76.45	76.14	76.75

The architectures mentioned above and their variants in terms of the number of layers were trained to JAFFE and CK+ datasets[1]. To a fair comparison, we have also included the encoder architecture of the CAE (Section 3.3). Except for the AlexNet and the CAEs, we have used the pre-trained weights on the ImageNet dataset, considering two different fine-tuning schemas. First, (1) all weights of convolutional layers are frozen, and only fully connected layers are fine-tuned to the FER problem. The second schema also fine-tunes the last convolutional layer (2).

The rationale for using pre-trained weights is based on the fact that FER datasets do not have sufficient data to fine-tune deep models correctly. Besides, this strategy is a common practice in state-of-the-art [48–51] and should provide meaningful insights when comparing the TL with the STL strategy.

Table 10 shows the performance of all experiments. The columns denoted with (1) and (2) represent the results using the respective fine-tuning schema for FER datasets.

[1] All trained models are available for research purposes at [the hyperlink will be inserted in the final version]

Table 8 Accuracy considering the LOSO protocol and a pool with 5 representations generated with the proposed method. LFW and CK+ are auxiliary and target datasets, respectively. FER models based on a pool of SVMs, RF, BG, and dynamic selection based on KnoraU in a collection of DTs and RF. Strategies S, L, and A represent different seeds, latent vectors, and CAE architectures. For each ensemble, the single best classifier and the following fusion strategies: sum, product, and stacking. (*) is the pool accuracy standard deviation. The best results are in bold.

Strategy		Pool of	Ensembles		Dinamic Selection (KnoraU)	
		SVMs	BG	RF	DT	RF
S	Best Single	83.24 (1.69)	66.75 (1.68)	63.22 (1.28)	66.96 (2.04)	64.77 (1.17)
	Sum	85.93	73.08	68.80	71.86	69.41
	Product	85.93	73.39	69.11	73.08	69.72
	Stacking	**87.15**	74.92	73.70	72.47	72.47
A	Best Single	84.91 (0.74)	66.82 (1.04)	65.38 (1.42)	67.20 (0.96)	66.37 (1.80)
	Sum	87.46	74.00	67.88	73.70	70.03
	Product	**88.07**	74.92	68.19	74.00	70.33
	Stacking	87.76	76.45	73.70	73.39	74.31
L	Best Single	84.27 (3.47)	68.62 (3.21)	64.26 (2.50)	69.15 (3.20)	64.50 (2.25)
	Sum	**87.76**	75.35	68.80	75.53	68.50
	Product	87.46	77.06	68.19	74.92	68.80
	Stacking	87.46	78.28	74.92	76.14	73.39
SA	Best Single	88.16 (3.35)	71.96 (3.05)	66.77 (1.56)	72.45 (3.01)	67.07 (2.36)
	Sum	87.76	72.17	70.64	71.55	71.55
	Product	87.76	73.08	70.64	71.55	71.55
	Stacking	**88.99**	74.61	75.84	71.55	74.31
SL	Best Single	85.69 (0.99)	68.75 (2.29)	66.96 (2.46)	68.77 (1.88)	68.26 (2.22)
	Sum	90.21	75.53	70.33	75.84	70.64
	Product	89.60	75.22	70.33	75.53	70.94
	Stacking	**90.51**	78.89	77.67	75.22	77.37
LA	Best Single	83.92 (1.42)	71.56 (2.68)	67.17 (1.30)	69.83 (2.02)	68.24 (1.72)
	Sum	87.15	72.47	70.94	73.08	70.33
	Product	86.85	73.39	70.33	72.78	70.33
	Stacking	**87.76**	76.75	77.06	71.86	75.22
SLA	Best Single	86.70 (5.41)	71.14 (4.09)	66.97 (2.69)	70.67 (3.90)	65.14 (1.86)
	Sum	86.54	74.92	69.41	73.08	69.41
	Product	85.62	75.22	69.72	74.61	69.41
	Stacking	**86.85**	74.61	73.70	73.70	74.00

Table 9 Ablation experiment of the β parameter considering the best setup of the proposed method for JAFFE and CK+ datasets.

	β	Accuracy (%)
JAFFE	0.0001	60.56
	0.001	**66.66**
	0.01	62.91
	0.1	60.56
CK	0.0001	91.13
	0.001	**92.66**
	0.01	89.60
	0.1	90.82

Table 10 CNN architectures trained with the LOSO protocol and two fine-tuning strategies (1, 2).

Architecture	CK(1)	CK(2)	JAFFE(1)	JAFFE(2)
VGG_16	78.9	46.8	40.6	14.1
VGG_19	81.4	61.1	44.5	14.5
InceptionV3	75.7	84.2	42.2	36.9
ResNet50	84.3	81.6	**56.7**	47.9
ResNet101	83.5	81.3	43.4	51.2
InceptionResnetV2	78.5	84.6	36.6	44.2
EfficientNetB0	80.6	**86.3**	38.3	49.0
EfficientNetB7	75.2	83.5	33.6	**52.6**
DenseNet121	81.1	81.1	54.9	45.4
DenseNet201	**85.8**	82.0	54.5	50.7
Encoder (CAE)*	69.3		39.3	
Alexnet*	70.6		42.9	
Ours	**92.66**		**66.66**	

(*)The model was trained from scratch (no pre-trained weights)

Finally, the models marked with an asterisk represent those trained from scratch once they had fewer layers to fine-tune.

The results allow us to draw some conclusions. The proposed REL surpassed pre-trained CNN models. The best CNN-based solution for the JAFFE dataset achieved 56.7% using ResNet50, while the proposed STL-based solution achieved 66.66% on the same dataset. A similar behavior was observed for the CK+ dataset. The best CNN solution achieved 86.3%, while the proposed STL-based method achieved an accuracy of 92.66%. It means that independently of the fine-tuning schema, the STL can surpass the TL by a fair margin, providing a better representation even when learning features from a different context.

The trade-off between classification time and accuracy is discussed when comparing STL and traditional supervised approaches. To clarify this, we evaluated the classification time for each test set using the LOSO protocol. The results are depicted in Figs. 8 and 9, for JAFFE and CK+ datasets, respectively. In such an analysis, we compare the best ensemble generated by the REL algorithm against traditional supervised learning approaches, previously discussed in Table 10. They are compared considering execution time during the test (classification time), accuracy, and model size in megabytes.

The proposed approach generally exhibits a better trade-off regarding classification time versus accuracy. A practical example can be seen in Fig. 8, where the RF-KnoraU approach outperforms the well-known networks VGG16, Inception, and ResNet. In Fig. 9, the result of the REL algorithm obtained using a pool of SVMs on the CK+ dataset outperforms CNN architectures in terms of accuracy and most of them in terms of classification time.

Finally, we present a comparative study with the results of our proposed model, which addresses the generation of a pool of unsupervised representations and its relationship with the most recent advances in facial emotion recognition, using the same

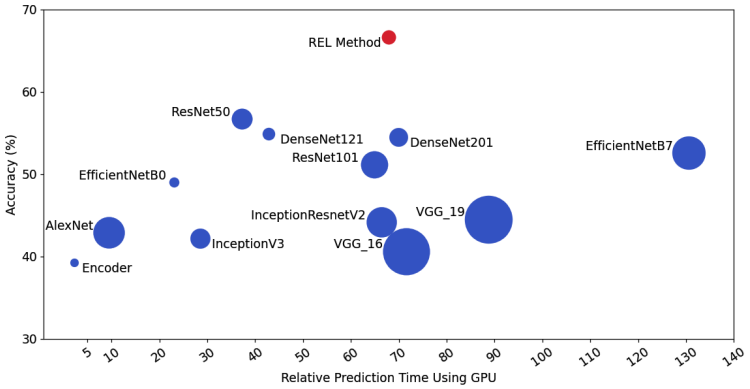

Fig. 8 Comparison of the proposed STL-based method (in red) with CNN architectures (in blue) in terms of accuracy (%), classification time (in seconds), and disk space (in megabytes) as the circle size. Results were computed using the JAFFE dataset and the LOSO protocol. The STL-based method considers the final classifier RF with KnoraU dynamic selection.

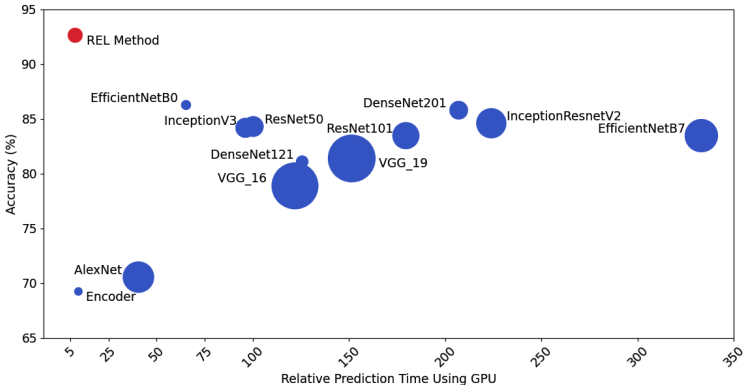

Fig. 9 Comparison of the proposed STL-based method (in red) with CNN architectures (in blue) in terms of accuracy (%), classification time (in seconds), and disk space (in megabytes) as the circle size. Results were computed using the CK+ dataset and the LOSO protocol. The STL-based method considers as the final classifier a pool of SVMs.

experimental protocol. This comparison aims to provide an expanded understanding of the contributions and improvements brought by our approach. Our effort focused on gathering the most significant possible number of studies that adhered to the same adopted evaluation protocol based on the LOSO protocol. Tables 11 and 12 present the highest accuracies observed in JAFFE and CK+ datasets, respectively, that have been documented in widely referenced scientific publications and conferences. The data revealed that the proposed method sets a new benchmark for facial expression recognition.

Table 11 Benchmark comparison on the JAFFE dataset using the LOSO protocol.

Reference	Method	Accuracy (%)	Classes	Feature Type
Kola & Samayamantula [52]	LGC-HD	60.70	6	Handcrafted
	LGC-HD	58.20	7	Handcrafted
Kartheek et al. [53]	RMP_Prime	61.64	6	Hand-crafted
Mandal et al. [54]	DRADAP	57.22	6	Handcrafted
	ARADAP	56.20	7	Handcrafted
Our Method		**66.66**	7	Deep

Table 12 Benchmark comparison on the CK+ dataset using the LOSO protocol.

Reference	Method	Accuracy (%)	Classes	Feature Type
Du & Hu [55]	WPLBP	91.72	6	Handcrafted
	WPLBP	86.47	7	Handcrafted
Wu & Lin [56]	GM	86.83	7	Deep
	GM+AFM	87.78	7	Deep
	GM+ W-AFM	88.25	7	Deep
	GM + W-CRAFM	89.84	7	Deep
Lee et al. [57]	CER-ICV	92.34	7	Handcrafted
Kola & Samayamantula [52]	LGC-HD	72.80	6	Handcrafted
	LGC-HD	70.60	7	Handcrafted
Our Method		**92.66**	7	Deep

Through these experiments, it becomes evident that generating a pool of unsupervised representations characterized by their diversity plays a fundamental role in improving the performance of an FER system.

4.5 Discussion

Concerning the auxiliary dataset, the experiments showed that the best results were found when the ensemble of representations was learned using the small Kyoto dataset. Moreover, such a dataset is far from the target domain, but it still provides better results when compared to those observed using the LFW dataset, which is composed of faces. In summary, our results indicated that the variability of images within the dataset is a more critical factor for performance than the total number of images or the proximity to the target problem. This finding suggests that the right choice for the auxiliary dataset must consider data diversity (image variability), which is essential for learning robust representations.

Based on the results observed in our experiments and analyses, we can positively answer our first research question, RQ1. The unsupervised representations with diversity obtained through the REL algorithm significantly impacted the performance of the proposed FER model. By incorporating diversity in an unsupervised manner through different initialization strategies and a custom loss function, the REL algorithm has demonstrated its effectiveness in producing a set of comprehensive representations. These representations capture latent and complex features in facial expressions, allowing the model to extract meaningful and distinctive features.

We obtained high accuracy in recognizing facial expressions by training the model with the unsupervised representations generated by the REL. The proposed method is up to 5.1% and 1.3% better than the single classifier for JAFFE and CK+ datasets, respectively.

The diversity introduced by REL made it possible for the model to become more adaptable to different variations in expressions, including nuances and subtle differences between categories of expressions. In addition, the generalizability of the model has been greatly improved. The diversified representations allowed the model to learn general characteristics that were not restricted to a specific set of training data, making it more efficient in identifying facial expressions in new datasets. This reduces the tendency to overfit by preventing the model from overspecializing on specific training data.

Furthermore, including a pool of unsupervised representations generated by the REL algorithm facilitated the knowledge transfer process. The pre-training of the unsupervised representations allowed the FER model to initialize with more comprehensive knowledge and, later to be adjusted for the specific FER task. This improved the overall performance of the model.

The answer to our second research question (RQ2) was only possible after the last experiments in which the proposed STL-based approach was compared favorably with the CNN-based FER solutions.

The comparative analysis between the proposed method and the CNN models adjusted for JAFFE and CK+ datasets revealed considerable improvements in terms of precision. Specifically, the proposed method outperformed the fitted CNN models by up to 9.9% in the JAFFE dataset and by up to 6.3% in the CK+ dataset. These substantial gains in accuracy are of great relevance and demonstrate the effectiveness of the STL-based approach.

A favorable comparison with supervisory-based CNN solutions suggests that the proposed STL-based approach can provide more promising and reliable results for Facial Expression Recognition. By eliminating the need for many labels in the early stages of training, the STL approach has proven to be efficient and cost-effective while achieving competitive results in terms of accuracy and generalization.

5 Conclusion

We have proposed a new method based on STL applied to the FER problem. The primary advancements within our self-taught learning approach, as compared to the existing literature, can be summarized as (a) the REL algorithm, which can generate a pool with diverse representations using an unsupervised dataset, and (b) a robust performance evaluation, drawing a direct comparison between our suggested unsupervised FER solution and supervised methods within identical conditions, this entails utilizing the same experimental protocol and addressing the challenges posed by a genuine and challenging task such as FER.

As future work, we plan to investigate different directions. First, we intend to evaluate an alternative scheme for the loss function employed while learning the unsupervised set of latent representations, aiming at amplifying the diversity. Second, we

plan to extend the evaluation of the proposed REL algorithm to cover different target problems, exploring its applicability and effectiveness in other contexts. In addition, we plan to consider the investigation of transformers and large vision models, especially to evaluate whether these architectures provide additional performance gains compared to those explored in our work. These initiatives aim to improve further the scope and robustness of our contributions.

Acknowledgement

We would like to thank the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq, grants 306878/2022-4, 406030/2023-5 and 441610/2023-4) and *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) for co-financing this research.

6 Declarations

This section summarizes some important statements as follows:

- **Funding and/or Conflicts of Interests/Competing Interests**: The authors certify that they have no conflict of interest.
- **Data Availability**: The datasets used in the experiments are available to the scientific community upon request and sign of a proper responsibility agreement.
- **Authorship Confirmation**: all authors have participated in the conception, design, analysis, and interpretation of the data, drafting the article or revising it, and approving the final version.

References

[1] Yeung, M.K.: A systematic review and meta-analysis of facial emotion recognition in autism spectrum disorder: The specificity of deficits and the role of task characteristics. Neuroscience & Biobehavioral Reviews **133**, 104518 (2022)

[2] Vehlen, A., Kellner, A., Normann, C., Heinrichs, M., Domes, G.: Reduced eye gaze during facial emotion recognition in chronic depression: Effects of intranasal oxytocin. Journal of Psychiatric Research **159**, 50–56 (2023)

[3] Qiao, Y., Zeng, K., Xu, L., Yin, X.: A smartphone-based driver fatigue detection using fusion of multiple real-time facial features. In: 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 230–235 (2016). IEEE

[4] Li, S., Deng, W.: Deep facial expression recognition: A survey. IEEE transactions on affective computing **13**(3), 1195–1215 (2020)

[5] Rathour, N., Singh, R., Gehlot, A., Akram, S.V., Thakur, A.K., Kumar, A.: The decadal perspective of facial emotion processing and recognition: A survey. Displays, 102330 (2022)

[6] Ganaie, M.A., Hu, M., Malik, A., Tanveer, M., Suganthan, P.: Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence **115**, 105151 (2022)

[7] Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th International Conference on Machine Learning, pp. 759–766 (2007)

[8] Bengio, Y., Bastien, F., Bergeron, A., Boulanger-Lewandowski, N., Chherawala, Y., Cisse, M., Côté, M., Erhan, D., Eustache, J., Glorot, X., *et al.*: Deep self-taught learning for handwritten character recognition. In: NIPS* 2010 Deep Learning and Unsupervised Feature Learning Workshop (2010)

[9] Ojala, T., Pietikäinen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern recognition **29**(1), 51–59 (1996)

[10] Zavaschi, T.H., Britto Jr, A.S., Oliveira, L.E., Koerich, A.L.: Fusion of feature sets and classifiers for facial expression recognition. Expert Systems with Applications **40**(2), 646–655 (2013)

[11] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**, 91–110 (2004)

[12] Haralick, R.M., Shanmugam, K., Dinstein, I.H.: Textural features for image classification. IEEE Transactions on systems, man, and cybernetics (6), 610–621 (1973)

[13] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Computer vision and image understanding **110**(3), 346–359 (2008)

[14] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 886–893 (2005). Ieee

[15] Canal, F.Z., Müller, T.R., Matias, J.C., Scotton, G.G., Sa Junior, A.R., Pozzebon, E., Sobieranski, A.C.: A survey on facial emotion recognition techniques: A state-of-the-art literature review. Information Sciences **582**, 593–617 (2022)

[16] Feng, S., Yu, H., Duarte, M.F.: Autoencoder based sample selection for self-taught learning. Knowledge-Based Systems **192**, 105343 (2020)

[17] Allognon, S.O.C., Britto, A.d.S., Koerich, A.L.: Continuous emotion recognition via deep convolutional autoencoder and support vector regressor. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020). IEEE

[18] Delazeri, B.R., Vera, L.L., Barddal, J.P., Koerich, A.L., *et al.*: Evaluation of self-taught learning-based representations for facial emotion recognition. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2022). IEEE

[19] Renda, A., Barsacchi, M., Bechini, A., Marcelloni, F.: Comparing ensemble strategies for deep learning: An application to facial expression recognition. Expert Systems with Applications **136**, 1–11 (2019)

[20] Wen, G., Hou, Z., Li, H., Li, D., Jiang, L., Xun, E.: Ensemble of deep neural networks with probability-based fusion for facial expression recognition. Cognitive Computation **9**(5), 597–610 (2017)

[21] Li, R., Ren, C., Zhang, X., Hu, B.: A novel ensemble learning method using multiple objective particle swarm optimization for subject-independent eeg-based emotion recognition. Computers in biology and medicine **140**, 105080 (2022)

[22] Dhankhar, P.: Resnet-50 and vgg-16 for recognizing facial emotions. International Journal of Innovations in Engineering and Technology (IJIET) **13**(4), 126–130 (2019)

[23] Chowdary, M.K., Nguyen, T.N., Hemanth, D.J.: Deep learning-based facial emotion recognition for human–computer interaction applications. Neural Computing and Applications **35**(32), 23311–23328 (2023)

[24] Akhand, M., Roy, S., Siddique, N., Kamal, M.A.S., Shimamura, T.: Facial emotion recognition using transfer learning in the deep cnn. Electronics **10**(9), 1036 (2021)

[25] Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Unsupervised learning of hierarchical representations with convolutional deep belief networks. Communications of the ACM **54**(10), 95–103 (2011)

[26] Markov, K., Matsui, T.: Music genre classification using self-taught learning via sparse coding. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1929–1932 (2012). IEEE

[27] Hu, H., Phan, N., Chun, S.A., Geller, J., Vo, H., Ye, X., Jin, R., Ding, K., Kenne, D., Dou, D.: An insight analysis and detection of drug-abuse risk behavior on twitter with self-taught deep learning. Computational Social Networks **6**(1), 1–19 (2019)

[28] Qureshi, A.S., Khan, A., Shamim, N., Durad, M.H.: Intrusion detection using deep sparse auto-encoder and self-taught learning. Neural Computing and Applications **32**, 3135–3147 (2020)

[29] Li, S., Li, K., Fu, Y.: Self-taught low-rank coding for visual learning. IEEE transactions on neural networks and learning systems **29**(3), 645–656 (2017)

[30] Ramamurthy, S.R., Ghosh, I., Gangopadhyay, A., Galik, E., Roy, N.: Star-lite: A light-weight scalable self-taught learning framework for older adults' activity recognition. Pervasive and Mobile Computing **87**, 101698 (2022)

[31] Germani, E., Fromont, E., Maumet, C.: On the benefits of self-taught learning for brain decoding. GigaScience **12**, 029 (2023)

[32] He, P., Jia, P., Qiao, S., Duan, S.: Self-taught learning based on sparse autoencoder for e-nose in wound infection detection. Sensors **17**(10), 2279 (2017)

[33] Liu, L., Wei, W., Chow, K.-H., Loper, M., Gursoy, E., Truex, S., Wu, Y.: Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In: 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 274–282 (2019). IEEE

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.*: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)

[35] Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with gabor wavelets. In: Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200–205 (1998). IEEE

[36] Lucey, P., Cohn, J.F., Kanade, T., Saragih, J., Ambadar, Z., Matthews, I.: The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: 2010 Ieee Computer Society Conference on Computer Vision and Pattern Recognition-workshops, pp. 94–101 (2010). IEEE

[37] Viola, P., Jones, M.J.: Robust real-time face detection. International journal of computer vision **57**, 137–154 (2004)

[38] Doi, E., Inui, T., Lee, T.-W., Wachtler, T., Sejnowski, T.J.: Spatiochromatic receptive field properties derived from information-theoretic analyses of cone mosaic responses to natural scenes. Neural computation **15**(2), 397–417 (2003)

[39] Uetz, R., Behnke, S.: Large-scale object recognition with cuda-accelerated hierarchical neural networks. In: 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 1, pp. 536–541 (2009). IEEE

[40] Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In: Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition (2008)

[41] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L.: Review of deep

learning: Concepts, cnn architectures, challenges, applications, future directions. Journal of big Data **8**, 1–74 (2021)

[42] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)

[43] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[44] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)

[45] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[46] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)

[47] Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114 (2019). PMLR

[48] Xu, M., Cheng, W., Zhao, Q., Ma, L., Xu, F.: Facial expression recognition based on transfer learning from deep convolutional networks. In: 2015 11th International Conference on Natural Computation (ICNC), pp. 702–708 (2015). IEEE

[49] Zhang, J., Li, W., Ogunbona, P., Xu, D.: Recent advances in transfer learning for cross-dataset visual recognition: A problem-oriented perspective. ACM Computing Surveys (CSUR) **52**(1), 1–38 (2019)

[50] Peng, M., Wu, Z., Zhang, Z., Chen, T.: From macro to micro expression recognition: Deep learning on small datasets using transfer learning. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 657–661 (2018). IEEE

[51] Shao, J., Qian, Y.: Three convolutional neural network models for facial expression recognition in the wild. Neurocomputing **355**, 82–92 (2019)

[52] Kola, D.G.R., Samayamantula, S.K.: Facial expression recognition using singular values and wavelet-based lgc-hd operator. IET Biometrics **10**(2), 207–218 (2021)

[53] Kartheek, M.N., Prasad, M.V., Bhukya, R.: Radial mesh pattern: a hand-crafted feature descriptor for facial expression recognition. Journal of Ambient Intelligence and Humanized Computing **14**(3), 1619–1631 (2023)

[54] Mandal, M., Verma, M., Mathur, S., Vipparthi, S.K., Murala, S., Kranthi Kumar, D.: Regional adaptive affinitive patterns (radap) with logical operators for facial expression recognition. IET Image Processing **13**(5), 850–861 (2019)

[55] Du, L., Hu, H.: Weighted patch-based manifold regularization dictionary pair learning model for facial expression recognition using iterative optimization classification strategy. Computer Vision and Image Understanding **186**, 13–24 (2019)

[56] Wu, B.-F., Lin, C.-H.: Adaptive feature mapping for customizing deep learning based facial expression recognition model. IEEE access **6**, 12451–12461 (2018)

[57] Lee, S.H., Baddar, W.J., Ro, Y.M.: Collaborative expression representation using peak expression and intra class variation face images for practical subject-independent emotion recognition in videos. Pattern Recognition **54**, 52–67 (2016)